# Using a Layered Approach for Interoperability on the Semantic Web

Isabel F. Cruz Huiyong Xiao
*Department of Computer Science*
*University of Illinois at Chicago*
{ifc | hxiao}@cs.uic.edu

## Abstract

*In this paper, we further develop a proposed layered approach for the Semantic Web. Our objective is to build a specific solution to the problem of providing data interoperability among different databases, so as to allow for schematic data integration. In particular, we solve the problem of translating queries on a database schema into queries on another database schema, using their relationship with an ontology. We use RDF Schema to model the databases and the ontology. A common vocabulary expresses the mappings between each database schema and the ontology.*

## 1. Introduction

The World Wide Web has been made possible on the basis of widely established standards. However, there are many intelligent services such as information brokers, search agents, and information filters that are limited in their functionality and only work as stand-alone services, which do not interoperate. The Semantic Web has been proposed to add semantics to web contents and make the Web machine understandable and processable, thus making interoperability a reality [14].

With the advent of XML, a syntactic platform has been created. However, data standardization in itself is not a complete solution [22]. The lack of well understood and agreed upon semantics has led to problems in integrating data from various sources and in achieving a seamless platform for web transactions. The Resource Description Framework, RDF, and its vocabulary description language, RDF Schema, create a layer on top of XML. Such a layer may well provide a common basis for expressing the necessary semantics to enable the interoperability between applications that exchange machine-understandable information on the Web.

Meanwhile, a number of web applications that have been introduced lately are aimed at providing integrated access to data and information. For many of these applications, data interoperation is essential. Six general levels of interoperability between two or more spatially distributed independent applications can be identified [7]. These levels range from network protocols and operating systems to data models and application semantics. For the lower levels, users need very specific knowledge such as the particular operating system being used. At the data model level, users are provided with a single virtual global data model, which is an abstraction of all the underlying remote databases. However, knowledge of the semantics of the remote databases is still required. At the semantics application level, seamless communication between remote databases is possible without prior knowledge of their underlying semantics.

Motivated by the layering principles used in networking, Melnik and Decker proposed a layered approach to solve the problem of interoperability of information models [18]. Their approach focuses on identifying three data modeling layers, the *syntax*, the *object* and the *semantic* layers, and associated modeling primitives. In this paper, we focus on the semantic layer and propose an approach for interoperability based on SQL-like queries, using the RQL query language for RDF [15]. The mapping between similar concepts is facilitated by a global ontology and by a common vocabulary. In particular, we focus on posing queries on one schema, the *source* or *local* schema, and mapping those queries to a second schema, the *target* or *remote* schema. In our approach, all the data models including the ontology are expressed using RDF Schema. In the future, we anticipate that we will be able to extend our framework to a more expressive language such as DAML+OIL, which is defined on top of the RDF primitives [14] and uses an additional inference mechanism.

Different kinds of heterogeneities are at the center of attaining data interoperability. Such heterogeneities can be classified as syntactic, schematic, or semantic [7]. We look at the particular problem of solving schematic heterogeneities, such as those that arise from using different schemas to represent the same data. We aim to encode schematic heterogeneities in the semantic layer to

Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)
0-7695-1999-7/03 $17.00 © 2003 IEEE

enable machine processing, thus relieving users from understanding the role played by the concepts in each of the data models.

The rest of this paper is organized as follows. Section 2 presents related work. An overview of the layered approach of Melnik and Decker and our own refinement of that approach for the semantic layer are presented in Section 3. In Section 4, we use a case study to illustrate the different steps in the query rewriting algorithm that maps a query on the source schema to an equivalent query on the target schema. Finally, directions for future work are presented in Section 5.

## 2. Related work

The subject of interoperability and the related subject of data integration have been long-standing in database research. In the nineties, several system architectures and systems have been proposed (e.g., TSIMMIS [12] and InfoHarness [24]). Recently, several fundamental issues have been addressed [9, 10]. A central problem in data integration is the schema or ontology mapping, for which there generally exist two approaches: Global-As-View (GAV) and Local-as-View (LAV) [9, 10]. In the GAV approach, every entity in the global schema is associated with a view over the source local schema. In the LAV approach, the local schemas are defined as views over the global schema. Although query processing is usually simpler in the GAV approach, the LAV approach is preferred in the situation where the system needs to be easily maintained and extended for the sake of the manipulations over new source local schemas. Our approach is based on the LAV approach. Specifically, we use an ontology with which we can establish connections between the different schemas, similarly to the role played by domain maps [16], but do not attempt to integrate all the database schemas into a single schema.

A number of recent proposals have produced considerable impact in the field of data integration. In particular, the MOMIS (Mediator Environment for Multiple Information Sources) system consists of a pool of tools for the integration of information in heterogeneous sources ranging from traditional databases to XML databases [3, 4, 5]. The integration process relies heavily on a common thesaurus derived from the WordNet lexical database. The inter-schema knowledge is expressed using the thesaurus in the form of relationships such as synonymy and hyponymy. One tool is the SI-Designer (Source Integrator Designer) [3, 4], a designer support tool for E-commerce applications, which uses a semi-automatic approach (i.e., requiring some user intervention) for the integration of heterogeneous data sources. Another tool is the MIKS (Mediator agent for Integration of Knowledge Sources) system [5], an agent-based middleware system that integrates data belonging to heterogeneous sources into a global virtual view and offers support for the execution of queries over the global virtual schema [8].

The Clio project creates mappings between two data representations semi-automatically for the purpose of managing and facilitating heterogeneous data transformation and integration [20]. Given the two schemas and the set of correspondences between them, Clio can generate queries (e.g., SQL, XSLT, XQuery) that drive the translation of data conforming to the source schema to data conforming to the target schema. However, apart from the mapping queries, Clio does not provide a mechanism for users to associate semantic information with the data, apart from the semantics that are "embedded" in the queries.

Another approach, which uses an ontology to achieve data integration in distributed databases in a geospatial application, is based on declarative agreements between the schemas of the local databases and the ontology [11, 13]. Such agreements effectively capture the semantics in the particular case of mappings between data values. End users can seamlessly query and aggregate semantically related data using a visual interface. An expert at the site of the local database uses another visual interface to specify the agreement between that database and the ontology.

In general, the issue of capturing the semantics of the data and of the mappings remains, therefore, an important goal. In this regard, the layered model for the Semantic Web of Melnik and Decker [18] can provide the foundation to attain that goal. They emphasize the advantages of using layers. For example, the complexity of data model interoperation is reduced to interoperation within a specific layer or sublayer. Also, comprehensive APIs for individual layers facilitate reuse and reduce the costs of application development, and implementations of the layers and agreements between peer layers can be modified without affecting the applications. In their work, they mainly identify the object layer that fills the gap between the syntax and semantic layers. The object layer includes key features such as identity and binary relationships, basic typing, reification, ordering, and n-ary relationships. They also examine design issues and implementation alternatives involved in building the object layer.

In our paper, we mainly concentrate on the semantic layer. By comparison with other existing approaches for data integration or interoperability [25], we list the advantages of our approach in comparison with the features of other approaches:

- **Ontology representation**: We use RDF and RDF Schema to uniformly express and model the semantic layer, including the global ontology, local schemas, and the mapping information (i.e.,

the common vocabulary) between a remote schema and the global ontology. This approach provides for an extensible basis for semantic-level interoperability.

- **Ontology role**: To describe the semantics of information sources, SIMS [2] uses a *single ontology approach*, in which all information sources are directly related to a shared global ontology. This approach requires that all sources provide nearly the same view on a domain. The OBSERVER system [19] uses a *multiple ontology approach*, in which each information source is described by its own ontology separately. It needs an additional representation formalism defining the inter-ontology mapping between each pair of separate ontologies. Our framework uses a *hybrid approach*, which overcomes some of the drawbacks of the above two approaches. There are two sorts of ontologies being used, namely the local unified ontology representing single or multiple related schemas in the local site and the global ontology representing the domain knowledge. The advantage of a hybrid approach is that new sources can easily be added without the need for modifying the mappings associated with the other sources or the shared vocabulary.
- **Query reformulation**: We propose a bidirectional query rewriting algorithm. In our approach we can translate a query in terms of the global ontology into sub-queries for each appropriate source, similarly to the query reformulation in SIMS [2]. We can also rewrite the query posed on any local schema into queries against all the other schemas, just like the query processing in the *peer-to-peer* system [23], in which users do not need to know the structure and the contents of the peer schemas and of the global ontology.

## 3. Overview

### 3.1. Layered Approach

In this section, we give an overview of the layered approach proposed by Melnik and Decker [18] and the architecture of a typical application based on this layered approach. Of the four proposed layers (see Figure 1), we concentrate on the semantic layer and identify three sublayers that are contained in it.

**3.1.1. Application Layer.** The application layer is used to express queries. For example, a visual user interface may be provided in this layer to allow users to submit their queries [13]. Ideally, query results are integrated and shown to the users, so as to give the appearance that the distributed databases interoperate seamlessly with each other.

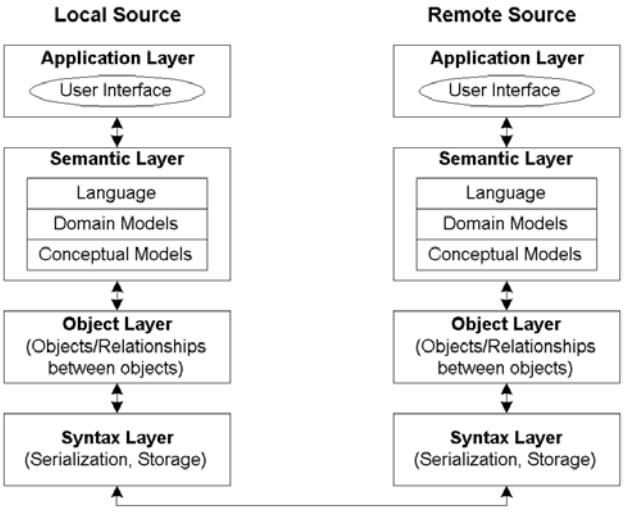



Figure 1. **Model of the layered approach**

**3.1.2. Semantic Layer.** This layer consists of three sub-layers, each of which accomplishes different tasks but cooperate to satisfy the purpose of data interoperability.

- **Languages:** The main purpose of this sublayer is to accept the user queries and to interpret them so that they can be understood by the other interoperating systems. The language can be highly specialized or be a general purpose language within the Semantic Web.
- **Domain Models:** They express the ontologies of a particular application domain, and can be different from one other even for the same domain [16]. Examples of domains include transportation, manufacturing, e-business, digital libraries, and aircraft maintenance.
- **Conceptual Models:** They model concepts, relationships and constraints using different types of constructs (e.g., generalization, aggregation, cardinality constraints). RDF Schema and UML Foundation/Core are two examples.

**3.1.3. Object Layer.** The purpose of the object layer is to give an object-oriented view of the data to the application. This layer enables manipulations of objects and binary relationships between them. Every object in a data schema is mapped to a particular class. The object layer also forwards all the information that it receives from the syntax layer to the semantic layer.

**3.1.4. Syntax Layer.** The main purpose of the syntax layer is to provide a way of specifying both the semantic and object layer information using a common

representation. XML has been used to represent RDF and RDF Schema. This layer is also responsible for serializing the data and for mapping the queries from the object layer to the XML representation of the data schemas. The information that is extracted by the queries is returned to the object layer.

### 3.2. Architecture

Based on the layered approach discussed above, we mainly concentrate on realizing the *semantic layer* to solve the data interoperability problem. The main task of the semantic layer is to accept user queries from the application layer, and process the query. Our approach uses RDF Schema for the conceptual model of the data sources and for the domain model. The latter expresses the global ontology for the application domain. Given our choice of RDF and RDF Schema, the RDF Query Language (RQL) [15] is the natural choice for the language sublayer whose role is to accept and interpret queries from users.

The application layer accepts RQL queries. As for the implementation of the object layer and of the syntax layer, we use RSSDB (The RDF Schema Specific Database), which is an RDF Store that uses schema knowledge to automatically generate an Object-Relational (SQL3) representation of RDF metadata, and load resource descriptions [1].

Figure 2 shows the architecture that can be used to build applications based on our layered approach to the data interoperability problem. The whole process can be divided into three sub-processes as follows:

- **Constructing local unified schemas:** A component called Schema Integrator is used to transform data schemas that exist in a local system automatically into a single schema, the *local unified schema*. We consider relational, XML, and RDF databases.
- **Mapping Process:** We use a global ontology expressed in RDF Schema to serve as the mediator between different local schemas, each of which is mapped to the global ontology. We utilize a common vocabulary, which is also expressed in RDF Schema, to facilitate this semi-automatic mapping process.

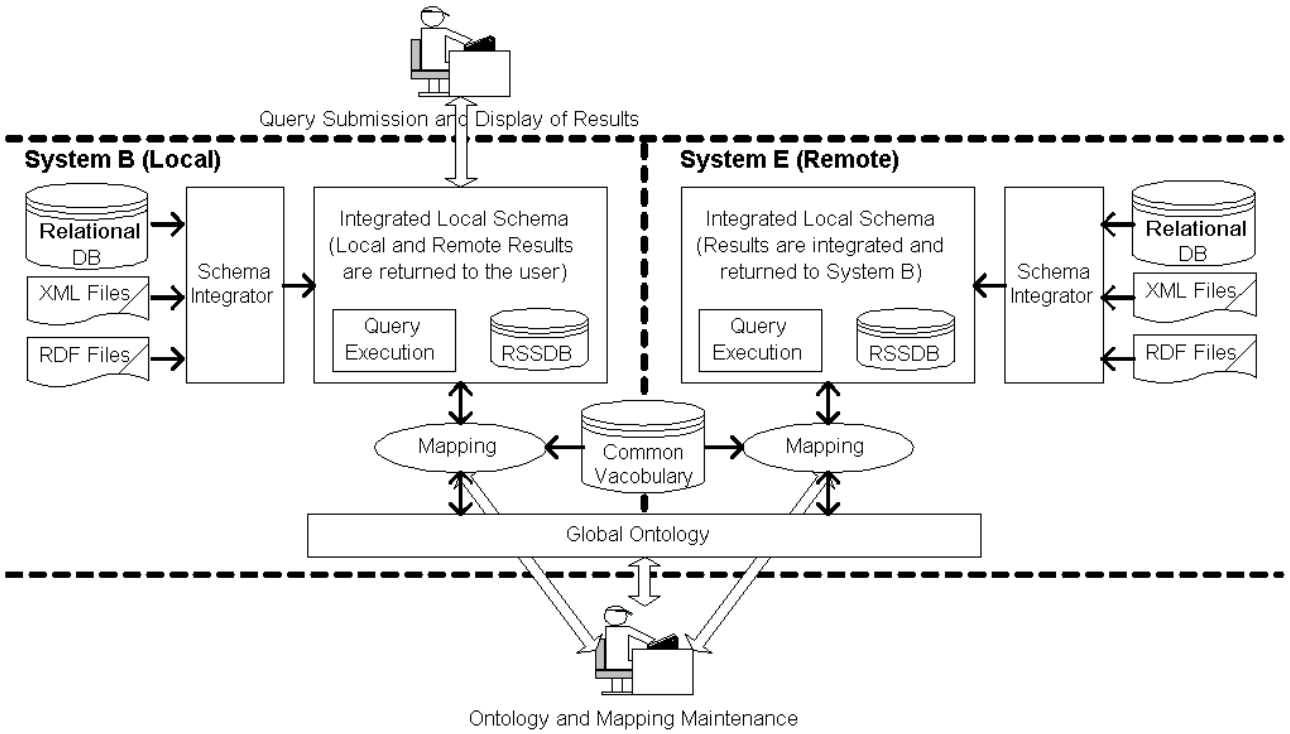



Figure 2. **Architecture for data interoperation**

- **Query Processing:** When the user submits a query against the local unified schema, the query will be executed directly over the local RSSDB. This process is called *local query processing*. Meanwhile, queries are transformed from one local unified schema to any other schema by the query rewriting algorithm (discussed later), so that all the resources matching the query will be retrieved. This process based on the mapping information between global ontology and local schemas is called *global query processing*, which can be performed automatically under some simplifying assumptions. The answers to both local query processing and global query processing are assembled and returned to the user.

# 4. The layered approach

## 4.1. A case study

Consider several legacy systems for aircraft maintenance scheduling with their own local schemas (adapted from [22]) that need to exchange data and share information that is stored in relational or XML databases (see Figure 3). Examples of schematic heterogeneities include the fact that the table name "RDYF15S" in System D, the attribute name "F15S" in System C, and the tuple value "F15" in System B are carrying the same information. Our approach aims to solve the data interoperability problem between different systems by hiding such schematic heterogeneities.

## 4.2. Construction of the local unified schema

The construction of the local unified schema includes two phases. In the first phase, we construct the local unified schema, represent it using RDF Schema, and store it in RSSDB. In the second phase, we transform data from the original relational database or XML files into RDF files, which are then also stored into RSSDB.

**System A**

***Table: ACMAINT***

| ACTYPE | RDYWHEN | NUM | STATE | BASENAME | MECHANIC |
|---|---|---|---|---|---|
| F15 | 0500 | 22 | CA | Anaheim | F15_team |
| F16 | 1700 | 16 | CA | Chico | F16_team |

**System B**

***Table: RDYACFT***

| MODEL | AVAILTIME | QTY | AIRBASE | STAFF_ID |
|---|---|---|---|---|
| F15 | 0800 | 12 | CA, Anaheim | 1214 |
| F16 | 1000 | 13 | GA, Dalton | 1215 |

***Table: STAFF***

| S_ID | TITLE | TEAM_LEADER | STAFF_NUM |
|---|---|---|---|
| 1214 | F15_team | Johnson | 6 |
| 1215 | F16_team | Michael | 5 |

**System C**

***Table: AIRCRAFT***

| RDYTIME | F15S | F16S |
|---|---|---|
| 0500 | 22 | - |
| 1700 | - | 16 |

***Table: MAINTSCHED***

| MECH_GROUP | AIRCFT |
|---|---|
| F15_group | F15S |
| F16_group | F16S |

**System D**

***Table: RDYF15S***

| WHEN | QUANTITY |
|---|---|
| 0900 | 22 |

***Table: RDYF16S***

| WHEN | QUANTITY |
|---|---|
| 1300 | 16 |

**System E**

**aircraft.dtd**

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT AIRCRAFT_SCHEDULE (AIRCRAFT)*>
<!ELEMENT AIRCRAFT (NUMBER, RDYTIME, AIRBASE,
                    MTSTAFF)+>
<!ATTLIST AIRCRAFT NAME CDATA #REQUIRED>
<!ELEMENT NUMBER (#PCDATA)>
<!ELEMENT RDYTIME (#PCDATA)>
<!ELEMENT AIRBASE (#PCDATA)>
<!ELEMENT MTSTAFF (#PCDATA)>
```

**aircraft.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AIRCRAFT_SCHEDULE SYSTEM "aircraft.dtd">
<AIRCRAFT_SCHEDULE>
        <AIRCRAFT NAME="F-15">
                <NUMBER> 5 </NUMBER>
                <RDYTIME> 11:00 am </RDYTIME>
                <AIRBASE> Anaheim, CA </AIRBASE>
                <MTSTAFF> Eagle-1 </MTSTAFF>
        </AIRCRAFT>
        <AIRCRAFT NAME="F-16">
                <NUMBER> 4 </NUMBER>
                <RDYTIME> 12:00 am </RDYTIME>
                <AIRBASE> Naples, FL </AIRBASE>
                <MTSTAFF> Tiger-1 </MTSTAFF>
        </AIRCRAFT>
</AIRCRAFT_SCHEDULE>
```

Figure 3. **Five aircraft maintenance legacy systems with different data schemas**

**4.2.1. Local unified schema.** When the underlying schemas are associated with tables in a relational database, we analyze the attributes of each table and the dependency relationships between every pair of tables through foreign keys.

If two tables are connected using a foreign key, we create a schema tree for each table. The root of each tree represents one table with the type being *rdfs:Class* and its children represent the attributes of the table with the type being *rdf:Property*. Therefore, the edges connecting the root and its children are *rdfs:domain* edges. Then we connect two trees using an *rdfs:range* edge from the node corresponding to the foreign key to the root of the other tree. Hence, we obtain the unified schema that can be used by the system for data interoperation. An example is shown in Figure 4 (a) for System B where *STAFF_ID* is a foreign key, and the local unified schema is shown in Figure 4 (a).

In the case of System D, where there is no foreign key between two tables, for each table we also create a schema. We then build a new root with type *rdfs:Class* and two new nodes of type *rdf:Property* as children of this new root. Then we connect each one of these nodes to one of the nodes representing the tables using an edge of type *rdfs:range*.

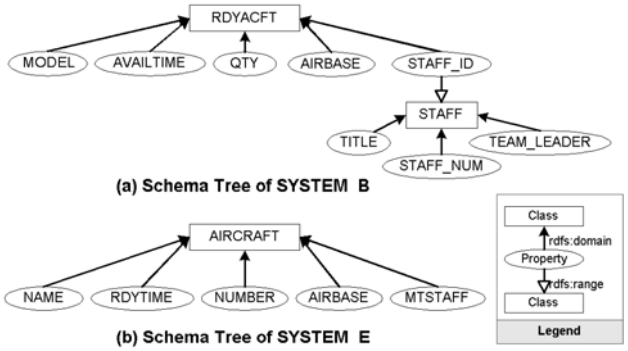



Figure 4. **Schema trees of the legacy systems**

If a system uses XML files, we analyze the DTD file corresponding to each XML file, and find all the element tags and attribute tags, and their hierarchy, which we use to construct the local unified schema tree for the system. Figure 4 (b) shows the schema tree for System E.

After all the participating systems have their local schemas organized into a single unified schema they are represented in an RDF Schema and stored in RSSDB. We use RSSDB to store the local integrated schema and its data so that local users can manipulate it using RQL and its APIs. Figure 5 gives a fragment of the underlying RDF Schema representation of the local unified schema of System E.

```
<rdfs:Class    rdf:ID = ”AIRCRAFT”>
</rdfs:Class>
<rdf:Property    rdf:ID = “NAME”>
   <rdfs:domain    rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property    rdf:ID = “NUMBER”>
   <rdfs:domain    rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Integer”/>
</rdf:Property>
<rdf:Property    rdf:ID = “RDYTIME”>
   <rdfs:domain    rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property    rdf:ID = “AIRBASE”>
   <rdfs:domain    rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property    rdf:ID = “MTSTAFF”>
   <rdfs:domain    rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
</rdf:Property>
```

Figure 5. **A fragment of RDF schema for the local unified schema of system E**

**4.2.2. Data transformation.** We consider two cases when transforming the data from a schema into a unified local schema.

**1) Relational Database.** We use the *left outer join* operation to unite the data from multiple tables that are connected through foreign keys. The table containing the foreign key acts as the left part of the *left outer join* operation. Figure 6 gives the example of a SQL query for such a transformation in System B. In the case that there is no foreign key relationship, we use a Cartesian product to realize the data transformation.

```
select R.MODEL, R.AVAILTIME, R.QTY, R.AIRBASE, S.TITLE,
       S.TEAM_LEADER, S.STAFF_NUM
from RDYACFT  R
left join STAFF  S
on R.STAFF_ID=S.S_ID
```

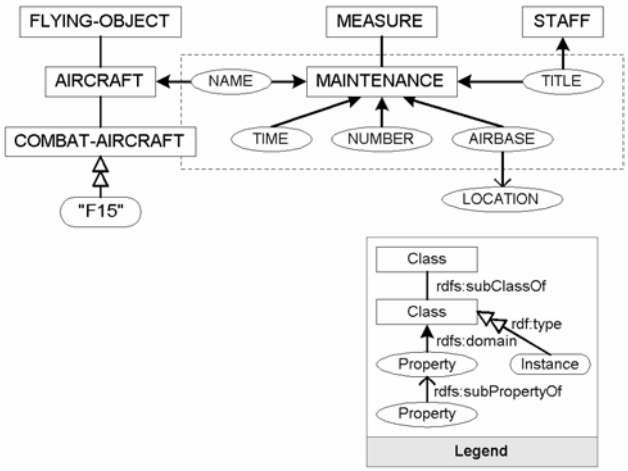
Figure 6. **SQL Query for transforming data in System B**

**2) XML files.** The data is organized either using XML Schema or based on DTD. In both situations we use XSLT expressions to transform data from an XML file into an RDF file. After the data is saved into an RDF file, we can use RQL or any of the APIs for RSSDB.

## 4.3. Mapping Process

**4.3.1. Global Ontology.** The global ontology is used for mediating among distributed schemas [16]. For this purpose, we set up the relationships between the global ontology and the database schemas based on a common vocabulary. In this sense, the common vocabulary contains the necessary mapping information (see the next subsection for details). Figure 7 shows a fragment of the global ontology for the domain of aircraft maintenance

and its partial RDF representation. We focus on the class and properties that are contained in the dashed line box. Notice that we use *rdfs:isDefinedBy*, which is an RDF property, for a description of a resource so as to specify the reference of each class or property to the common vocabulary.




```
<rdfs:Class rdf:ID="Maintenance">
   <rdfs:subClassOf rdfs:Resource="#Measure"/>
</rdfs:Class>
……
<rdf:Property rdf:ID="Name">
   <rdfs:domain rdfs:Resource="#Aircraft"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Aircraft"/>
</rdf:Property>
<rdf:Property rdf:ID="Time">
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Date"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_ReadyTime"/>
</rdf:Property>
<rdf:Property rdf:ID="Number">
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Integer"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Number"/>
</rdf:Property>
<rdf:Property rdf:ID="Airbase">
   <rdfs:subPropertyOf rdfs:Resource="#Location"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Address"/>
</rdf:Property>
<rdf:Property rdf:ID="Title">
   <rdfs:domain rdfs:Resource="#Staff"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Machinist"/>
</rdf:Property>
……
```

Figure 7. **A fragment of the global ontology and its partial RDF Schema description**

**4.3.2. Common vocabulary.** An important component of our approach is that each schema shares a dictionary or thesaurus with the global ontology (see Figure 8). This dictionary stores the common vocabulary of all the schema concepts and the relationships between the ontology and each schema. When presented with a new schema that needs to be mapped to the global ontology, the system checks every concept name against the dictionary to obtain an appropriate matching for that concept, and chooses the optimal one according to a *Scoring Function*. The construction of a dictionary and the determination of an appropriate matching was discussed in detail elsewhere [6, 17, 21], therefore we will not elaborate any further on this issue. In this paper, we consider that the dictionary has a simplified structure, which only supports one-to-one total mappings. In order to extend our approach, other semantic criteria and a more general structure for the dictionary will be considered in the future.

AIRCRAFT – Flighttype, Actype, Model, AC_type, FlightModel, F<number>S, Aeroplane, F-<number>, F_<number>, Airplane, Model_Flight

NUMBER – Num, Qty, Quantity, Num_ready, ReadyNumber, Many, Size, Rdynumber, QuantityReady, Rdyqty

READYTIME – Rdywhen, Availtime, When, RdyTime, ReadyTime, FinishRepair, Ready, CompletionTime, PickupTime, DeliveryTime

ADDRESS – LocationRdywhen, AirportAddr, Airbase, Location, AirportLocation

MACHINIST – Craftsman, Mechanic, Repairman, Technician, Staff

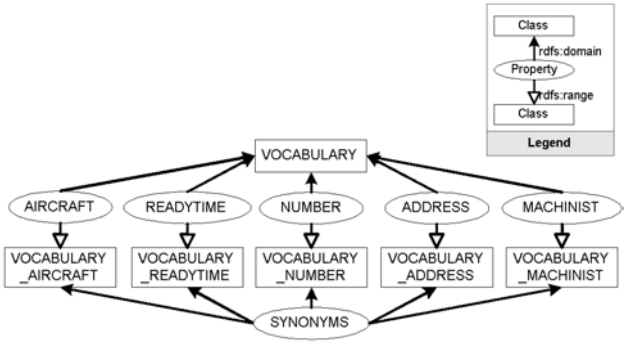



Figure 8. **A fragment of the common vocabulary in the thesaurus**

**4.3.3. Mapping.** So far the components that participate in the mapping process (i.e., the local unified schema, the common vocabulary, and the global ontology) are all represented using RDF Schema. The mapping between the global ontology and a local unified schema is realized according to the common vocabulary. Figure 9 shows the local unified schemas of the five legacy systems, the relationships among the local unified schemas, the global ontology, and the common vocabulary.

Each local unified schema and the global ontology have their properties mapped to the properties in the common vocabulary through *rdfs:isDefinedBy*. We can also see from Figure 7 how *rdfs:isDefinedBy* is used in the RDF representation of the global ontology. When the local unified schema becomes related to the common

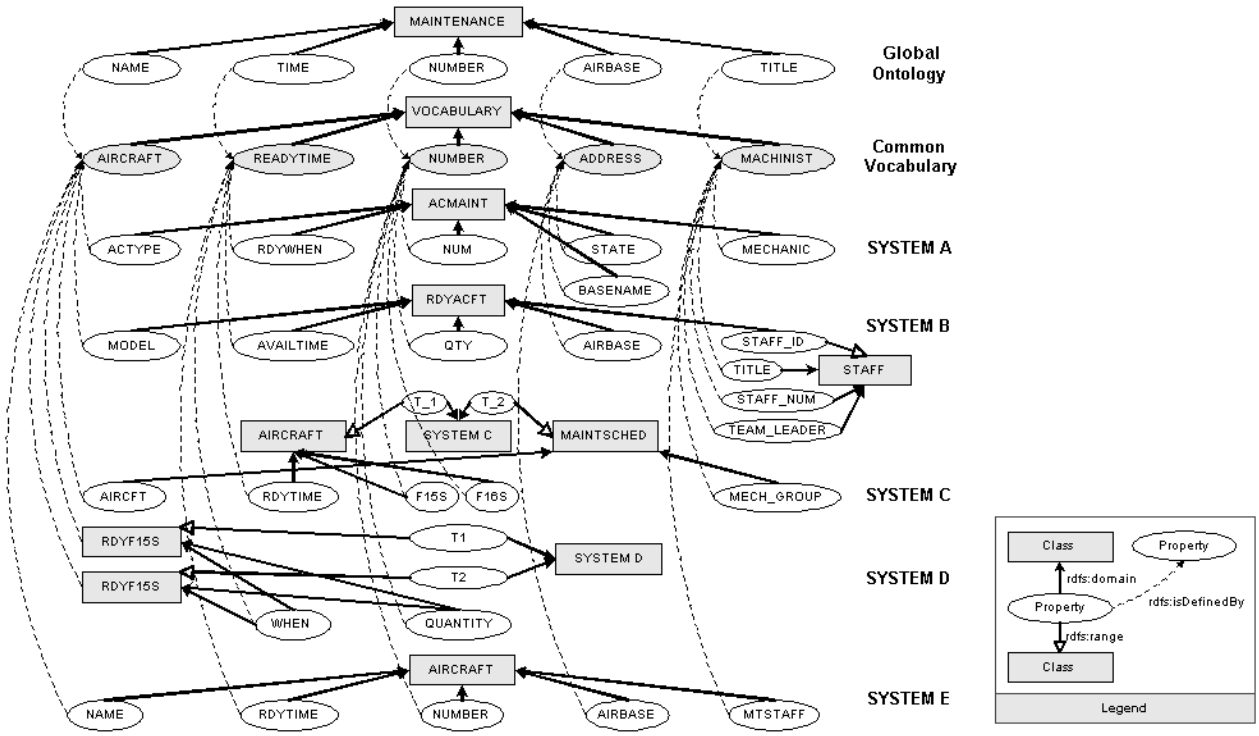



Figure 9. **Mapping between the global ontology and the database schemas**

vocabulary, the RDF representation of the local unified schema also needs to be updated by inserting *rdfs:isDefinedBy* into any *rdfs:Class* or *rdfs:Property* being mapped. We use System E as an example (refer to Figure 5 for the previous RDF description of its local unified schema). Figure 10 shows a fragment of the new RDF representation.

```
<rdfs:Class   rdf:ID = ”AIRCRAFT”>
</rdfs:Class>
<rdf:Property   rdf:ID = “NAME”>
   <rdfs:domain   rdf:resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
   <rdfs:isDefinedBy rdf:resource = “#Vocabulary_Aircraft”/>
</rdf:Property>
……
<rdf:Property    rdf:ID = “MTSTAFF”>
   <rdfs:domain   rdf:resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
   <rdfs:isDefinedBy rdf:resource = “#Vocabulary_Machinist”/>
</rdf:Property>
```

Figure 10. **RDF Schema of System E with mapping information**

## 4.4. Query Processing

**4.4.1. RQL.** In our approach, we choose to use the RDF Query Language (RQL) to express and interpret the user's query against the RDF resources, that is, the local unified schemas constructed previously, which exist in the distributed databases. RQL is a typed functional language and relies on a formal model for directed labeled graphs permitting the interpretation of superimposed resource descriptions by means of one or more RDF schemas [15]. In Figure 11, we show the example of an RQL query on the local unified schema in System B.

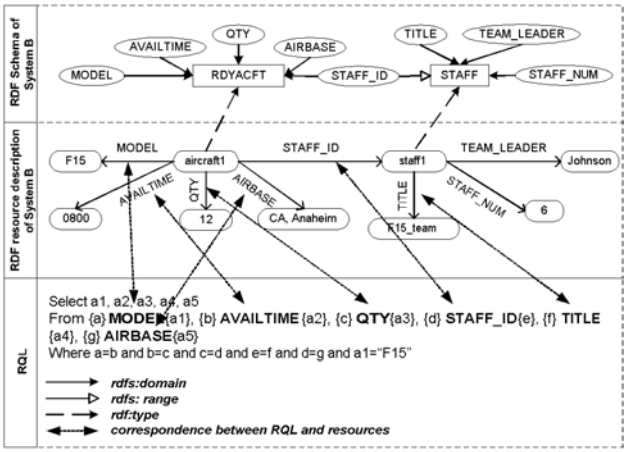



Figure 11. **A typical RQL query on schema of System B**

After the system executes the RQL query on the local unified schema and the resources are stored in the local RSSDB, the answer to the query is expressed in RDF and returned to the user (a visual user interface can display the results in the format that is most convenient to a particular user). Figure 12 shows the answer as expressed in RDF. This is the *local query processing* phase accomplished within the local site. In addition, to obtain

all the matching RDF resources residing in other sites within the network of databases, the query has to be rewritten against the RDF Schemas that describe those resources, a process we call *remote query processing*. In the next subsection, we will illustrate the query rewriting algorithm using an example.

```
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Bag>
  <rdf:Seq>
    <rdf:li rdf:type="string">F15</rdf:li>
    <rdf:li rdf:type="string">12</rdf:li>
    <rdf:li rdf:type="string">0800</rdf:li>
    <rdf:li rdf:type="string">CA, Anaheim</rdf:li>
    <rdf:li rdf:type="string">F15_team</rdf:li>
  </rdf:Seq>
</rdf:Bag>
</RDF>
```

Figure 12. **Query results of executing an RQL query on local System B**

**4.4.2. Query rewriting algorithm.** Our algorithm makes the following simplifying assumptions about the schemas, mappings, and queries: (1) We assume the local unified schema is either a hierarchy or can be converted into a hierarchy without losing important schematic or semantic information. Examples of hierarchies that were obtained from relational databases and from XML databases were shown in Figure 4. (2) We assume the mappings between the global ontology and local unified schemas are total mappings, i.e., all the concepts (classes or properties) occurring in the query are mapped. (3) We consider only one-to-one-mappings, that is, a concept in one schema maps to a single concept in another schema. (4) To keep the current discussion simple, we assume the variables in the query only refer to "pure" classes or properties, i.e., no functions such as *subClassOf*, *subPropertyOf*, *domain*, and *range* are applied on them, and we consider that queries are in the form of $\{class_1\}property_1\{class_2\}$, as in Figure 11. (5) We consider only schema mapping, not value mapping (which was the focus of [13]).

The rewriting algorithm utilizes the established mapping information between any pair of local unified schemas with the global ontology as the mediation. Before the algorithm starts, we should initialize the source local unified schema and the target local unified schema as schema trees by using the following rules. For every pair of concepts (classes or properties), say *S* and *O*, if *S rdfs:domain O*, or *S rdfs:subClassOf O*, or *S rdfs:subPropertyOf O*, we make *S* a child of *O*; if *S rdfs:range O* or *S rdf:type O*, we incorporate *S* and *O* into a single node. In addition, we establish mappings between the source schema and the target schema according to their respective mappings to the global ontology. Figure 13 shows the schema trees of System B and System E and their mappings (see also their local unified schema and mappings in Figure 9), which we will use to illustrate our algorithm later. In System B, for instance, the properties such as MODEL, AVAILTIME, and QTY are made children of the class RDYACFT. The property STAFF_ID and the class STAFF are incorporated into a single node.

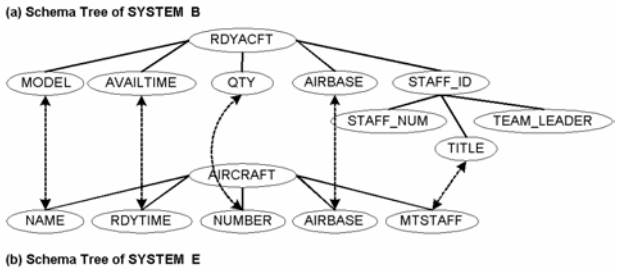



Figure 13. **Schema trees of System B and System E and their mappings**

In the following query rewriting algorithm, *RQL* is the source query and *RQL'* is the target query, and we use the RDF terminology of *subject*, *object*, and *statement* [15]. As an example, we consider the *RQL* query on the local unified schema of System B that is shown in Figure 11. *RQL'* is the target query on the unified schema in System E. The following steps will be executed:

**Step 1:** Get all the *statements* existing in the *from clause* which corresponds to the *objects* in the *select clause*. In our example, we obtain (***MODEL, AVAILTIME, QTY, TITLE, AIRBASE***). Each of these *statements* corresponds to a class or property in the source schema.

**Step 2:** Use the mapping information between the source schema and the target schema to find all the concepts (classes or properties) that correspond to the *statements* found in Step 1, and make these concepts statements in *RQL'*, as follows:

***select*** $<object_1>,...,<object_n>$
***from*** $\{<subject_1>\}$ $<statement_1>$ $\{<object_1>\}$, ..., $\{<subject_n>\}$ $<statement_n>$ $\{<object_n>\}$.

In our particular example, we have:

***select*** $o_1, o_2, o_3, o_4, o_5$
***from*** $\{s_1\}$***NAME***$\{o_1\}$, $\{s_2\}$***RDYTIME***$\{o_2\}$,$\{s_3\}$***NUMBER***$\{o_3\}$,$\{s_4\}$ ***AIRBASE***$\{o_4\}$,$\{s_5\}$***MTSTAFF***$\{o_5\}$

**Step 3:** Consider each pair of nodes, $E_i$ and $E_j$, corresponding to each pair of immediate *statements* in *RQL'*, for the following situations.

1) If $E_i$ and $E_j$ are siblings (have the same parent), then this pair of *statements* share a common *subject*, so we append the condition $<subject_i>=<subject_j>$ to the *where clause* of *RQL'* using *and*. For example, in System E (see Figure 13), ***NAME*** and ***RDYTIME*** are siblings, therefore, $s_1 = s_2$ is appended to the *where clause*.

2) If $E_i$ and $E_j$ have a common ancestor $E_0'$ that is not the parent, as shown in Figure 14(a), then we append all the intermediate nodes (i.e., $statement_k$), between $E_0'$ and $E_i$ and between $E_0'$ and $E_j$, to the *from clause* in the form $\{<subject_k>\}statement_k\{<object_k>\}$. In addition, we append new conditions to the *where clause* in the following way:

- <$object_1'$> = <$subject_2'$>, ..., <$object_k'$> = <$subject_{k+1}'$>, ..., <$object_{n-1}'$> = <$subject_n'$>, and <$object_n'$> = <$subject_i$>, which correspond to the path $E_0'$, $E_1'$, ..., $E_n'$, $E_i$.
- <$object_1''$> = <$subject_2''$>, ..., <$object_k''$> = <$subject_{k+1}''$>, ..., <$object_{n-1}''$> = <$subject_n''$>, and <$object_n''$> = <$subject_j$>, which correspond to the path $E_0'$, $E_1''$, ..., $E_n''$, $E_j$.
- < $subject_1'$> = <$subject_1''$>, since $E_1'$ and $E_1''$ share a common *subject*.

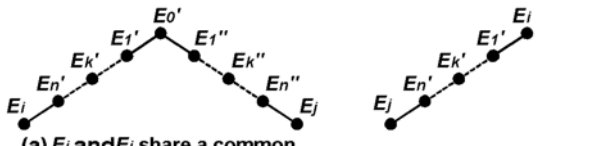



Figure 14. **Relationships between $E_i$ and $E_j$**

3) If $E_i$ is an ancestor of $E_j$ as shown in Figure 14(b), then we append all the intermediate nodes (i.e., $statement_k$) between $E_i$ and $E_j$ to the *from clause* in the form *{<$subject_k$>} $statement_k$ {<$object_k$>}*. In addition, we append <$object_i$> = <$subject_1'$>, <$object_1'$> = <$subject_2'$> ..., <$object_k'$> = <$subject_{k+1}'$>..., <$object_{n-1}'$> = <$subject_n'$>, and <$object_n'$> = <$subject_i$> to the *where clause*.

In our example, we only have the case of siblings. Therefore, after Step 3 we get the following *RQL'* query:

***select*** $o_1$, $o_2$, $o_3$, $o_4$, $o_5$
***from*** {$s_1$}**NAME**{$o_1$}, {$s_2$}**RDYTIME**{$o_2$},{$s_3$}**NUMBER**{$o_3$},{$s_4$}
**AIRBASE**{$o_4$},{$s_5$}**MTSTAFF**{$o_5$}
***where*** $s_1=s_2$ *and* $s_2=s_3$ *and* $s_3=s_4$ *and* $s_4=s_5$

**Step 4:** For each query condition of the form $o_i$=<*string-value*> in *RQL*, we append the condition $o_i'$=<*string-value*>" to the *where clause* of *RQL'*, where $o_i$ in *RQL* is mapped to $o_i'$ in *RQL'*,. In our example, we finally obtain the following *RQL'* query:

***select*** $o_1$, $o_2$, $o_3$, $o_4$, $o_5$
***from*** {$s_1$}**NAME**{$o_1$}, {$s_2$}**RDYTIME**{$o_2$},{$s_3$}**NUMBER**{$o_3$},{$s_4$}
**AIRBASE**{$o_4$},{$s_5$}**MTSTAFF**{$o_5$}".
***where*** $s_1=s_2$ *and* $s_2=s_3$ *and* $s_3=s_4$ *and* $s_4=s_5$ *and* $o_1$="F15"

**Step 5:** Finally, the *RQL'* query is executed on RSSDB in System E. For our running example, we obtain the answer that is shown in Figure 15, where *rdf:Seq* is a tuple.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Bag>
  <rdf:Seq>
    <rdf:li rdf:type="string">F-15</rdf:li>
    <rdf:li rdf:type="string">5</rdf:li>
    <rdf:li rdf:type="string">11:00 am</rdf:li>
    <rdf:li rdf:type="string">Anaheim, CA</rdf:li>
    <rdf:li rdf:type="string">Eagle-1</rdf:li>
  </rdf:Seq>
</rdf:Bag>
</RDF>
```

Figure 15. **Query results of executing an RQL query on remote System E**

This answer, which is returned by the *remote query processing* associated with System E will be assembled by unioning all the tuples with the *local query processing* results that were shown in Figure 12.

## 5. Conclusions and Future Work

In investigating a solution to the data interoperability problem, we further develop the layered approach proposed by Melnik and Decker [18]. Our approach is intended for the design and implementation of Semantic Web applications and uses RDF Schema as a common formalism. In particular, the design process includes three phases:

- The construction of the local unified schema at local sites, which is a fully automatic phase. Currently, we are able to unify relational tables, XML/DTD documents and local RDF sources.
- The creation of the global ontology, which is built by experts for particular application domains.
- The semi-automatic mapping process facilitated by a common vocabulary component.

We also propose a preliminary query rewriting algorithm for the query processing between schemas and ontologies modeled using RDF Schema. This process of query translation can be performed automatically.

Future work will focus on the following aspects: (1) We will lift some of the restrictions we imposed on the query rewriting algorithm. In particular, the algorithm will be extended from processing queries over tree-like ontologies to processing queries over graph-like ontologies. (2) We will further use the powerful querying capabilities of RQL on resource descriptions and schemas for the query translation across ontologies that are related by mappings other than one-to-one mappings, that is, one-to-many, many-to-many, and partial (one-to-null or null-to-one) mappings. We will also address mappings between concepts related by subclassing and the translation of queries containing *generalized path expressions*, featuring variables on labels for both classes and properties [15]. (3) We will concentrate on the creation of local unified schemas in order to make it general. (4) We will consider the case where the answers to the queries may be expressed in the native schema of the legacy databases. For example, by using RDF, the nesting structure associated with XML is lost in the mapping from XML data to RDF data. We will look at which properties can be encoded using RDF Schema, and how they can be encoded. If such information can be carried in the semantic layer then data interoperation among different data models can be dealt with. While this is a difficult problem in general, we believe that substantial contributions can be made by extending our current work in this direction. (5) We will look at the problem of extending vocabularies so that it can be more

suitable for the acquisition, representation, and execution of the ontology mappings in various cases. In fact, much work remains to be done in this area and in particular on semi-automatic methods to build the necessary correspondences with limited human intervention.

## Acknowledgments

We would like to thank Anjli Chaudhry, Gomathy Sundaresan, and Artan Alickolli, for their contributions. This work was partially supported by NSF Awards EIA-0091489 and ITR IIS-0326284.

## References

[1] S. Alexaki, N. Athanasi, V. Christophides, G. Karvounarakis, A. Maganaraki, D. Plexousakis, and K. Tolle, "The ICS-FORTH RDFSuite: High-level Scalable Tools for the Semantic Web". In *11th International World Wide Web Conference (WWW)*, 2002.

[2] Y. Arens, C. Hsu, and C. A. Knoblock, "Query processing in the SIMS information mediator". *Advanced Planning Technology, A. Tate (ed.), AAAI Press*, 1996.

[3] I. Benetti, D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini, "SI-Designer: an Integration Framework for E-commerce". In IJCAI Workshop on E-Business and the Intelligent Web, 2001.

[4] D. Beneventano, S. Bergamaschi, I. Benetti, A. Corni, F. Guerra, and G. Malvezzi, "SI-Designer: a tool for intelligent integration of information". In *34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, Volume 9, pp. 9088, 2001.

[5] D. Beneventano, S. Bergamaschi, G. Gelati, F. Guerra, and M. Vincini, "MIKS: An Agent Framework Supporting Information Access and Integration". *Intelligent Information Agents Research and Development in Europe: An AgentLink Perspective*, S. Bergamaschi, M. Klusch, P. Edwards, and P. Petta (eds.), LNCS, Springer Verlag, Volume 2586, pp. 22-49, 2003.

[6] J. Berlin and A. Motro, "Database Schema Matching using Machine Learning with Feature Selection". In *14th Intl. Conference on Advanced Information Systems Engineering (CAiSE),* pp. 452-466, 2002.

[7] Y. Bishr, "Overcoming the semantic and other barriers to GIS interoperability". *International Journal of Geographical Information Science 12(4)*: 299-314, 1998.

[8] G. Cabri, F. Guerra, M. Vincini, S. Bergamaschi, L. Leonardi, and F. Zambonelli, "Momis: Exploiting Agents to Support Information Integration". *IJCIS* 11(3):293-314, 2002.

[9] A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini, "On the Expressive Power of Data Integration Systems". In *21st Intl. Conference on Conceptual Modeling (ER)*, pp. 338-350, 2002.

[10] A. Calì, G. D. Giacomo, and M. Lenzerini, "Models for information integration: turning local-as-view into global-as-view". In *10th Intl. Workshop on Foundations of Models for Information Integration,* 2001.

[11] P. Calnan and I. F. Cruz, "Object Interoperability for Geospatial Applications". In *1st Semantic Web Working Symposium*, pp. 229-243, 2001.

[12] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "The TSIMMIS Project: Integration of heterogeneous information sources". In *16th Meeting of the Information Processing Society of Japan*, pp. 7-18, 1994.

[13] I. F. Cruz, A. Rajendran, W. Sunna, and N. Wiegand, "Handling Semantic Heterogeneities Using Declarative Agreements". In *10th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS),* pp. 168 –174, 2000.

[14] S. Decker, S. Melnik, F. V. Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The Semantic Web: The Roles of XML and RDF". *IEEE Internet Computing* 4(5): 63-74, 2000.

[15] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl, "RQL: A Declarative Query Language for RDF". In *11th International World Wide Web Conference (WWW)*, 2002.

[16] B. Ludäscher, A. Gupta, and M. Martone, "Model-Based Mediation with Domain Maps". In *17th Intl. Conference on Data Engineering (ICDE)*, Heidelberg, Germany, IEEE Computer Society, pp. 81-90, 2001.

[17] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid". In *27th Intl. Conference on Very Large Databases* (*VLDB),* pp. 49-58, 2001.

[18] S. Melnik and S. Decker, "A Layered Approach to Information Modeling and Interoperability on the Web". In *ECDL'00 Workshop on the Semantic Web*, 2000.

[19] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi, "Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies". In *1st IFCIS International Conference on Cooperative Information Systems (CoopIS)*, pp. 14-25, 1996.

[20] R. J. Miller, L. M. Haas, and M. A. Hernández, "Schema Mapping as Query Discovery". In *26th Intl. Conference on Very Large Databases* (*VLDB)*, pp. 77-88, 2000.

[21] E. Rahm, and P. A. Bernstein, "On Matching Schemas Automatically". *Microsoft Research Technical Report MSR-TR-2001-17*, February 2001.

[22] S. A. Renner, A. S. Rosenthal, and J. G. Scarano, "Data Interoperability: Standardization or Mediation". In *1st IEEE Metadata Conference*, 1996.

[23] O. D. Sahin, A. Gupta, D. Agrawal, and A. El Abbadi, "Query Processing Over Peer-To-Peer Data Sharing Systems". *Technical Report UCSB/CSD-2002-28,* University of California at Santa Barbara, 2002.

[24] L. Shklar, A. Sheth, V. Kashyap, and K. Shah, "InfoHarness: Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information". In *7th Intl. Conference on Advanced Information Systems Engineering (CAiSE)*, pp. 217-230, 1995.

[25] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner, "Ontology-Based Integration of Information - A Survey of Existing Approaches". In *IJCAI-01 Workshop: Ontologies and Information Sharing,* pp. 108-117, 2001.